03

Método Super

Transcripción

[00:00] Habíamos ya definido que dependiendo del tipo de funcionario que está aquí declarado, íbamos a calcular cuál sería su bonificación correcta, y nuestro código ya estaba funcionando, ya está calculado correctamente, pero también habíamos llegado a la conclusión de que este código no era escalable.

[00:24] Es muy trabajoso y muy desordenado continuar con esta estructura de código, de ifs entrelazados, y sobre todo definiendo tipos solamente al aire, como números cualquiera que pueden significar cualquier cosa, es una pésima práctica, nunca lo hagan.

[00:46] Entonces vamos a hacer este método un poco más dinámico, porque el funcionario tiene su bonificación y el gerente tiene su bonificación. Entonces, ya que el gerente tiene su bonificación, la cual es esta de acá, ¿qué pasaría si yo hago esto de aquí? Voy al gerente y escribo aquí public double getBonificación, abro llaves y retorno this.salario.

[01:28] Primer punto que voy a ver. No está compilando salario y ustedes se preguntarán: No es posible, pero si yo estoy extendiendo de funcionario y funcionario tiene salario, de hecho aquí está, yo lo veo. ¿Y por qué es que no compila? ¿Por qué no puedo acceder a salario? ¿La herencia no funciona o qué? Tranquilos, vamos a ver bien cómo funciona esto.

[01:55] Primero que todo, salario es una variable privada. Vamos a ver qué nos dice Eclypse. Nos dice: "El campo funcionario.salario no es visible". ¿Qué significa que no sea visible? Que al ser declarado aquí el funcionario como

privado simplemente él no es accesible con this. Recordemos qué hacía la palabra reservada this.

[02:25] This hacía referencia a este objeto, acuérdense, es this object, este objeto. Y este objeto no tiene salario, así de simple. Es por eso que no compila. ¿Qué soluciones tenemos al respecto? Bueno, podemos cambiarlo a public, lo cual sabemos que es una pésima práctica. ¿Resuelve? Sí, resuelve. Pero no lo vamos a hacer. ¿Por qué? Porque es una pésima práctica.

[03:03] No podemos tener a una variable pública si no tiene que ser pública. Recuerden eso: las variables son públicas solo cuando tienen que ser públicas. Si no tiene que ser pública o es privada o es protected, que es otro modificador de acceso, este de aquí, que si bien soluciona nuestro problema también, tampoco es la idea, porque protected es visible para la clase actual y para las clases hijas.

[03:39] Y gerente es una clase hija, entonces en ese caso protected sí es visible. ¿Pero cuál es el problema de protected? Que también es visible para las demás clases en el mismo paquete. Y esto ya nos trae un problema. Paquete es esto de aquí y quizás hasta ahora nosotros estamos agrupando todo en un mismo paquete y eso no está del todo bien. Pero como estamos yendo de menos a más, de a pocos, vamos a dejarlo en el mismo paquete por ahora pero no vamos a usar protected.

[04:14] Vamos a ponerle igual private. Protected es solo un spoiler que les estoy dando de lo que se va a venir más adelante en el curso. Lo vamos a ver de todas formas porque es muy importante, pero aún no es el momento. Tranquilos, ya va a llegar el momento. Paciencia. Y la pregunta continúa: ¿Cómo hacemos este código, compilar y funcionar, este código de aquí?

[04:38] Porque no está compilando en este momento. Si yo guardo esto de aquí, automáticamente él deja de compilar porque no es visible. ¿Cómo podría hacer

en ese caso? Ya que este caso es digamos conocido, Java también nos tiene una palabra reservada y es la palabra super.

[05:08] Con super nosotros tenemos acceso a los métodos de la clase padre. ¿Y adivinen qué? Podemos hacer un super.getSalario. ¿Qué significa super? Super es la palabra reservada diciéndonos: "Oye, quiero que de la clase padre, la clase que está arriba, de la clase superior, de la clase que está arriba de mí, llama a este método de él". Entonces así yo obtengo la bonificación correcta de mi gerente.

[05:51] En el caso de funcionario yo ya puedo limpiar este código diciéndole que ya retorné el 10% del salario. Entonces le doy aquí, borro esto de aquí, borro esto ya que no lo necesito, vamos a corregir un poco la indentación. Listo. Ya está el código ordenado y todo. Y ahora ya el funcionario normal ya tiene su bonificación del 10% y el gerente tiene su propia bonificación que va a ser el salario que yo le haya definido.

[06:28] ¿Será que va a funcionar? Vamos a ver a las clases de prueba, y si se dan cuenta yo no he movido nada, ni de aquí, ni de aquí, pero el código sigue compilando tranquilamente, no sucede absolutamente nada. Incluso aquí yo le voy a dar un gerente getBonificación. Nuevamente olvidé el detalle, voy a poner eso dentro de un syso. Ah, ya estaba aquí, lo había olvidado.

[07:04] Bueno, entonces vamos a ejecutarlo. Lo vamos a ejecutar y bonificación de 6000, bonificación de gerente correcta. Vamos a TestFuncionario, vamos a imprimir el salario y la bonificación de funcionario aquí. Damos a play y los mismos resultados, entonces hasta ahora todo está correcto.

[07:25] Nosotros tenemos los cálculos correctos y viendo el código, estamos viendo cómo poco a poco cada uno, cada entidad, digamos funcionario y gerente, van moldeando su propio comportamiento y están relacionadas conceptualmente, ya que gerente es un funcionario pero funcionario no es un gerente. Tienen que verlo de esa forma.

[07:52] ¿Todos los gerentes son funcionarios? Sí. Y entonces es un extend. Ahora, ¿todos los funcionarios son gerentes? No. Entonces no hay relación de vuelta, la relación queda ahí.